University of Reading

Implementation of deep neural networks (DNN) with batch normalization for batik pattern recognition

Article

Published Version

Creative Commons: Attribution-Share Alike 4.0

Open Access

Nurhaida, I., Ayumi, V., Fitrianah, D., Zen, R. A.M., Noprisson, H. and Wei, H. (2020) Implementation of deep neural networks (DNN) with batch normalization for batik pattern recognition. International Journal of Electrical and Computer Engineering (IJECE), 10 (2). pp. 2045-2053. ISSN 2088-8708 doi: https://doi.org/10.11591/ijece.v10i2.pp2045-2053 Available at https://centaur.reading.ac.uk/95532/

It is advisable to refer to the publisher's version if you intend to cite from the work. See Guidance on citing.

Identification Number/DOI: https://doi.org/10.11591/ijece.v10i2.pp2045-2053 <https://doi.org/10.11591/ijece.v10i2.pp2045-2053>

Publisher: IAES

All outputs in CentAUR are protected by Intellectual Property Rights law, including copyright law. Copyright and IPR is retained by the creators or other copyright holders. Terms and conditions for use of this material are defined in the End User Agreement.

www.reading.ac.uk/centaur

University of Reading

CentAUR

Central Archive at the University of Reading

Reading's research outputs online

International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 2, April 2020, pp. 2045~2053
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i2.pp2045-2053

Implementation of deep neural networks (DNN) with batch normalization for batik pattern recognition

Ida Nurhaida[1], Vina Ayumi[2], Devi Fitrianah[3], Remmy A. M. Zen[4], Handrie Noprisson[5], Hong Wei[6]
[1,2,3,5] Faculty of Computer Science, Universitas Mercu Buana, Indonesia
[4]School of Computing, National University of Singapore, Singapore
[5]Department of Computer Science, University of Reading, Reading RG6 6AY, United Kingdom

Article Info

Article history:

Received Jun 7, 2019
Revised Oct 22, 2019
Accepted Nov 1, 2019

Keywords:

Batch normalization
Deep learning
Deep neural network
Indonesian batik
Pattern recognition

ABSTRACT

One of the most famous cultural heritages in Indonesia is batik. Batik is a specially made drawing cloth by writing *Malam* (wax) on the cloth, then processed in a certain way. The diversity of motifs both in Indonesia and the allied countries raises new research topics in the field of information technology, both for conservation, storage, publication and the creation of new batik motifs. In computer science research area, studies about Batik pattern have been done by researchers and some algorithms have been successfully applied in Batik pattern recognition. This study was focused on Batik motif recognition using texture fusion feature which is Gabor, Log-Gabor, and GLCM; and using PCA feature reduction to improve the classification accuracy and reduce the computational time. To improve the accuracy, we proposed a Deep Neural Network model to recognise batik pattern and used batch normalisation as a regularises to generalise the model and to reduce time complexity. From the experiments, the feature extraction, selection, and reduction gave better accuracy than the raw dataset. The feature selection and reduction also reduce time complexity. The DNN+BN significantly improve the accuracy of the classification model from 65.36% to 83.15%. BN as a regularization has successfully made the model more general, hence improve the accuracy of the model. The parameters tuning also improved accuracy from 83.15% to 85.57%.

Copyright © 2020 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Ida Nurhaida,
Faculty of Computer Science,
Universitas Mercu Buana,
Jalan Raya Meruya Selatan no. 1, Kembangan Jakarta Barat-16550, Indonesia.
Email: ida.nurhaida@mercubuana.ac.id

1. INTRODUCTION

Technology and globalization have brought fundamental changes in the information technology [1]. Indonesia has a tangible and intangible cultural heritage. One of the most famous cultural heritages is batik. Citing the definition of the Indonesian Dictionary (KBBI), it is explained that batik is a specially made drawing cloth by writing *Malam* (wax) on the cloth, then processed in a certain way [2]. The word Batik originated from the Javanese language and composed of two parts, namely "Mbat" and "Titik", and means to make a "titik" (dot) [3]. Each region in Indonesia has different batik motifs and pattern rules [4]. The diversity of motifs both in Indonesia and the allied countries raises new research topics in the field of computer science, both for conservation, storage, publication and the creation of new batik motifs.

In computer science research area, studies about Batik pattern have been done by researchers and some algorithms have been successfully applied in Batik pattern recognition [5]–[7]. In previous work, Sanabila and Manurung (2009) conducted research on batik motifs recognition using template matching approach by applying the Generalized Hough Transform and key block frames [5]. At the same time,

Rahadianti, Manurung, and Pure (2009) researched the field of information retrieval using K-mean algorithm with feature extraction using Log Gabor filter and Color Histogram for batik image clustering [6]. In 2012, Nurhaida, Manurung, & Arymurthi conducted research on batik image dataset to compare performance of three methods such as Canny edge detection, grey level co-occurrence matrices (GLCM) and Gabor filter [7]. This study shows that using GLCM as features has performed the highest classification accuracy. Rangkuti *et al*., proposed Batik Image Retrieval based on similarity of shape and texture characteristics [8], and in 2014, they proposed Content-based Batik Image Retrieval based on shape and texture feature applying edge detection and wavelet transform method [9]. In 2014 Minarno *et al*. proposed Batik Image Retrieval based on enhanced micro-structure descriptor [10] and Batik Image Classification using co-occurrence matrices for extracting texture feature [11]. In 2015, Nurhaida *et al*. introduced an approach to batik pattern recognition using SIFT as a feature extraction method [12]. Furthermore, Fahmi *et al*. (in 2016), was conducted the Batik Image Retrieval using feature selection and reduction. This research shows that the selection and reduction feature process could improve the precision and reduce the execution time. From the experiments shows that PCA feature reduction can improve the retrieval precision while SFFS can reduce the execution time [13]. In 2017, Nurhaida *et al*. developed Texture Fusion for Batik Motif Retrieval System, they systematically investigates the impact of image texture features on batik motif retrieval performance [14]. Based on the result of earlier work [7] and [13], this study was focused in batik motif recognition using texture fusion feature which is Gabor, Log-Gabor, and GLCM and using PCA feature reduction to improve the classification accuracy and reduce the computational time.

On the other hand, recent research [14] did not use a deep learning approach. Study in Batik using Deep Learning was performed by [15], they proposed Deep Convolutional Network Transfer Learning for Batik Classification. Based on [15]–[18], deep learning can improve accuracy in pattern recognition. One of Deep Learning approaches is a Deep Neural Network (DNN). Deep neural network (DNN) is a powerful model that can perform high achievement on pattern recognition [19]. Even though DNN has a good reputation for solving a variety of pattern recognition task, training DNN is quite challenging [20]. Several results have appeared to challenge the implementation of DNN [20]–[23] One of the challenges is due to DNN training complicated because of the distribution of each layer's inputs changes during the training since the parameters of the previous layer are changed [24]. This method can slow down the training process due to requiring lower learning rates and careful parameter initialisation. This problem addressed by normalising the layer inputs for each training mini-batch. This normalisation method called Batch Normalization (BN) allows us to use higher learning rates and careless about initialisation. Furthermore, BN also acts as a regularizer and often eliminates the required of Dropout for regularisation [25]. This study proposed to used Deep Learning method, namely Deep Neural Network to make comparison with the previous study ([7] and [13]) and used batch normalisation as regularisation in our model. DNN with Batch Normalization is expected to make the model more general so that it can improve the accuracy of the model.

2. LITERATURE REVIEW

2.1. Batik

Indonesian Dictionary (KBBI) explains that batik is a specially made drawing cloth by writing the *Malam* (wax) on the cloth, then processing in a certain way. In Indonesia, each region has certain characteristics of the batik motifs also the way the motifs are organised [4]. The distinctiveness of batik motifs could be analysed from its characteristics of texture and shape, and the dissimilar motif has a dissimilar shape pattern. The Batik motifs pattern are divided into two groups namely geometric and non-geometric pattern. Groups of geometric pattern tend to have elements of symmetry that form triangle patterns, cross lines, rectangles, stars, parallelogram, and other patterns formed from the order of the line. The non-geometric pattern is a motif that has irregular composition of ornament like animals, plants, and so on. There are various types of geometric batik patterns; they are *Ceplok, Kawung, Lereng, Parang*, and *Nitik*. The motif of *Ceplok* has repetitive in geometric ornaments based on squares, circular shape, star, and other geometric shapes. The motif of *Kawung* known as the oldest batik pattern has the repetition of circles or elliptical shapes. The motif of *Lereng* has diagonal pattern lines that filled with small patterns. The motif of *parang* consists of parallel lines in diagonal shape filled with small ornaments. The motif of *Nitik* is made with small dots and lines that imitate the original woven fabric [3].

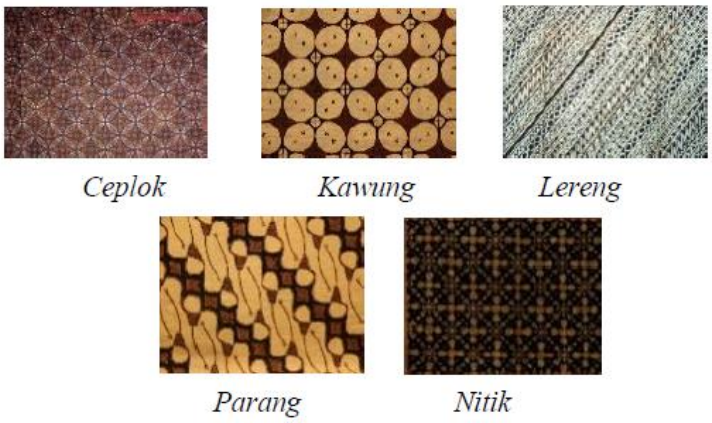

Figure 1. The instances of batik motif [12]

2.2. Deep neural network

Deep Neural Network is a type of Feed-Forward Neural Network that has more than one hidden layer between the input layer and the output layer. Every hidden layer j usually use a function of logistic to map the input from the preceding layer into scalar y_j which is then sent to the next layer.

$$y_j = logistic(x_j) = \frac{1}{1+e^{-x_j}}, \qquad x_j = b_j + \sum_i y_i w_{ij}$$

Where b_j is the bias of unit j, i is a unit index from the preceding layer and w_{ij} is the weight from every connection from unit j to unit i. To classify multi-class, the output unit j converted total of input x_i to the probability of class p_j by using non-linear softmax,

$$p_j = \frac{exp(x_j)}{\sum_k exp(x_k)}$$

where k is the class index.

DNN can be trained discriminatively by conducting backpropagation derivatives of a cost function which measure the difference of the target and actual output to obtain the result of training case [9]. When using softmax on output function, natural cost function C is cross-entropy between target probability d and softmax output p,

$$C = -\sum_j d_j \, log p_j,$$

Where the probability of a target is usually 1 to 0, it is the supervised information provided to train the DNN classifier [18].

2.3. Batch normalization

In conventional deep networks, excessive learning rates can cause gradients that explode or vanish and get stuck in poor local minima. The Batch Normalization helps solve these problems, by normalizing the activation through the network, so it prevents minor changes in parameters from reinforcement when data is propagated through the deep networks. Furthermore, the advantages of Batch Normalization causes the training more resilient to parameter scale. Typically, a large learning rate can escalate the scale of the layer parameters, which is afterwards amplifying the gradient throughout backpropagation and cause the model explosion, thus slowing down the training process. Still, with Batch Normalization, backpropagation through layers is not affected by its parameters scale [24]. Moreover, the BN even act as a regularises where in some cases can remove the need for Dropout.

Improving training performance can be done by fixing the distribution of the input layer during the training. The way to fasten the network training convergent is by whitening its inputs layer, for example, linear transformation to obtain unit variances and zero means, and decorrelated. However, to complete the network layer to be whitened needs expensive computation process. It happens because each layer that

observes the inputs generated by the layers below must have the same whitening of the inputs of each layer. Hence, to tackle this problem can be utilised by Batch Normalization.

Batch normalisation utilises the current mini-batch statistic by standardising the intermediate representations to approximate the layer whitening. A mini-batch is denoted as x, in which variance of each feature f along the mini-batch axis and the sample mean is calculated, and the size of the mini-batch is denoted as n, as follows.

$$\bar{x}_f = \frac{1}{n}\sum_{i=1}^{n} x_{i,f}$$

$$\sigma_f^2 = \frac{1}{n}\sum_{i=1}^{n}(x_{i,f} - \bar{x}_f)^2$$

By using this equation, each feature can be formulated as follows

$$\hat{x}_f = \frac{x_f - \bar{x}_f}{\sqrt{\sigma_f^2 + \epsilon}}$$

Where ϵ is a small positive constant to complete numerical stability enhancement; however, standardising activation of intermediate can diminish the representation of the layer power. To tackle this issue, BN proposes further learnable parameters γ and β, which individually shift and scale the data, pointing to a layer of the form.

$$BN(x_f) = \gamma_f \hat{x}_f + \beta_f$$

The original layer representation can be recovered by a network by setting γ_f to be σ_f and β_f to be $\bar{x}_f$ as follows.

$$y = \phi(W_x + b)$$

Where the matrix of weights is denoted as W, the vector of bias is denoted as b, the input of the layer is denoted as x and function of arbitrary activation is denoted as ϕ. Moreover, the BN is formalised as follows:

$$y = \phi(BN(W_x))$$

The vector of bias has been eliminated since its impact is cancelled by the standardisation. The method of backpropagation needs to be adjusted to generate gradients through the variance and mean computations as well after the normalisation has been the part of the network [26].

3. RESEARCH METHOD

The proposed method of this study consist of data preprocessing including feature extraction, fusion, and normalisation; feature reduction process; tuning hyperparameters; model building and testing; and evaluation. The diagram of research methodology showed in Figure 2. The dataset that we used in this study is batik image dataset that contained basic motif template form a certain class. The dataset contains 5 class; they are *Ceplok, Kawung, Lereng, Parang,* and *Nitik* [13]. In data preprocessing, the texture features have been extracted for every image; they are Gabor filters, log Gabor filters, GLCM, and LBP. From the four feature vectors then fusion and normalised to produce one feature vector. The feature reduction experiment is conducted to improve accuracy score and to reduce execution time rather than using all features. We divided the dataset into training and testing data and performed cross-validation. After that, we conducted the hyperparameters tuning to find the parameters value, and using these parameters value in our deep learning model. Then we conduct the model building, testing, and evaluate the model.

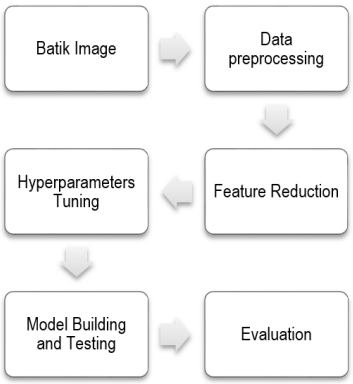

Figure 2. Research methodology

4. RESULTS AND ANALYSIS

4.1. Experimental setup

This study was conducted in Ubuntu 14.04 LTS 64-bit on a PC with Processor Intel® Core™ i7-6500U CPU @ 2.50GHz × 4, Memory DDR2 RAM 8 GB and Hard Disk 160 GB. In this study, we used Matlab to extracted, fusion, and normalised the four texture feature (Gabor filters, log Gabor filters, GLCM, and LBP). In our experiment, we divided the dataset into training and testing data and performed cross-validation. The training data took 70% of the dataset while the testing data took the rest. We implemented experiments using Keras deep learning library in Python. The Deep Learning method that we implemented is four layers Deep Neural Network (DNN) with Batch Normalization.

4.2. Feature extraction and fusion

In this study, we used four texture features as proposed [13]; they are Gabor filters, Log Gabor filters, GLCM, and LBP. The process of feature extraction and fusion successfully produced a feature vector with a length of 217 for each image. After the process of feature extraction and fusion, we conducted the feature reduction using PCA.

4.3. Features reduction

In this experiment, we conducted features reduction using PCA to reduce the component features. To make an equal comparison, we used the same batch_size and epoch value the batch_size value is 32 and epoch is 50. We used PCA component 25, 50, 100, 150, 200, 210, and 217 to select the best component of the feature vector. Table 1 gives the result of features reduction using PCA. Table 1 shows that the best accuracy achieved when the number of components (N) is 50. From this experiment of feature selection and reduction, we can produce the new feature vector and reduce the vector feature to length 50. This result shows that we can decrease the computational time, and gives more accuracy rather than using all features. Table 2 shows the report of classification in each class for the N= 50. As Table 2 presented, the best score in precision, recall, and f1-score is class 1. It is due to class 1 has more number of samples than other class, so that the learning is done well.

Table 1. Evaluation score of experiments using PCA feature reduction

Evaluation score	All	N=25	N=50	N=100	N = 150	N = 200	N = 210
Accuracy	0.8189	0.7884	0.8315	0.8010	0.8031	0.8168	0.8221
F1-Score	0.8171	0.7884	0.8288	0.7971	0.8008	0.8131	0.8188
Precision	0.8222	0.7961	0.8295	0.7985	0.8041	0.8154	0.8200
Recall	0.8189	0.7884	0.8315	0.8010	0.8031	0.8168	0.8221
Kappa	0.7368	0.6952	0.7562	0.7085	0.7125	0.7330	0.7421
Time (s)	58.345	48.933	49.236	52.464	55.350	58.265	58.018

Table 2. Evaluation score of each class for the N=50

Class	Precision	Recall	F1-score	support
1	0.88	0.91	0.89	448
2	0.68	0.54	0.60	72
3	0.80	0.72	0.76	97
4	0.77	0.87	0.82	186
5	0.84	0.76	0.80	147
avg	0.83	0.83	0.83	950

4.4. Hyperparameters tuning

In this scenario of the experiment, we used parallel processing to tuned the hyperparameters. As a result of PCA features reduction, in this scenario, we used the 50 features component that achieved in PCA features reduction. To tuned the hyperparameters in parallel processing, we used Grid Search algorithm that provided in the GridSearchCV class in Scikit-learn. We imported KerasClassifier and GridSearchCV in our code, to allow us to use Sklearn's Grid Search. The hyperparameters that we tuned they are: batch_size, epoch, learning rate, training optimisation algorithm, network weight initialisation, and neuron activation function.

1. Batch size and epoch

The result in tuning batch_size and epoch shows in Table 3. The result gives the accuracy of 79.13% with standard deviation score is 0.0116 using the number of batch_size 20 and epoch 200, while the computational time took 305.104926 seconds.

2. Learning rate

The result in tuning learning rate presented in Table 4. The result shows the best accuracy is 0.7895 when the leaning rate 0.001. The computational time in this process took 279.603224 seconds.

3. Optimiser algorithm

The result of the experiment in tuning optimiser algorithm presented in Table 5. This experiment gives the best accuracy of 78.50% with standard deviation score 0.012 when using RMSProp optimiser algorithm. The processing time in this experiment took 260.960422 seconds.

4. Network weight initialisation

Table 6 presented the result of tuning network weight initialisation. The result shows that lecun_uniform achieved the best accuracy of 75.51% with standard deviation 0.020530 and the computational time took 302.18661 seconds.

5. Neuron activation function

Table 7 presented the result of tuning neuron activation function experiment. The best result accuracy is achieved at 75.97% when using the linear function. The computational time to process this scenario took 298.757153 seconds.

Table 3. The accuracy score and standard deviation of batch size and epoch tuning

Epoch/ Batch_size	Epoch = 50	Epoch = 100	Epoch = 200
Bs = 20	0.777326 (0.006093)	0.774616 (0.015264)	**0.791328 (0.011656)**
Bs = 32	0.751129 (0.012336)	0.762873 (0.006160)	0.766938 (0.020129)
Bs = 40	0.748871 (0.013655)	0.763776 (0.021576)	0.772358 (0.013414)

Table 4. The accuracy score and standard deviation of learning rate tuning

Learning rate	Accuracy
0.001	0.789521 (0.017988)
0.01	0.783198 (0.016706)
0.1	0.626468 (0.015659)

Table 5. The accuracy score and standard deviation of optimiser algorithm tuning

Optimiser	Accuracy
SGD	0.758356 (0.001690)
RMSProp	0.785005 (0.012823)
Adagrad	0.777326 (0.014608)
Adadelta	0.771906 (0.010280)
Adam	0.776423 (0.015330)
Adamax	0.769648 (0.007978)
Nadam	0.777326 (0.004606)

Table 6. The accuracy score and standard deviation of network weight initialisation tuning

Unit_mode	Accuracy
Uniform	0.794490 (0.020530)
lecun_uniform	0.782746 (0.015502)
Normal	0.776874 (0.021262)
Zero	0.768744 (0.006760)
glorot_normal	0.782294 (0.007198)
glorot_uniform	0.778681 (0.007531)
he_normal	0.786360 (0.009539)
he_uniform	0.778681 (0.006093)

Table 7. The accuracy score and standard deviation of neuron activation tuning

Activation function	Accuracy
Softmax	0.789521 (0.022411)
Softplus	0.791780 (0.017988)
Softsign	0.786811 (0.016197)
Relu	0.771906 (0.015502)
tanh	0.776423 (0.015489)
Sigmoid	0.777778 (0.003319)
hard_sigmoid	0.787263 (0.002927)
linear	0.782746 (0.016791)

4.5. Model building, testing, and evaluation

In this stage, we build and evaluated our model (DNN+Batch Normalization) using Batik dataset. As a comparison, we also evaluate DNN without BN using Batik dataset. Table 8 gives the comparison of evaluation score of DNN and DNN+BN. The evaluation score shows that DNN+BN significantly improve the accuracy of the classification model from 65.36% to 83.15%. BN as a regularisation has successfully made the model more general, hence improve the accuracy of the model. We also implemented all parameters value that gave best accuracy from the experiment above, they are; batch_size = 20, epoch = 200, learning_rate = 0.001, optimizer = RMSprop, network weight initialization = Uniform, and neuron activation function = Softplus to our DNN model with batch normalization. Table 9 shows the evaluation score between DNN+BN Before tuning parameters and DNN+BN After tuning parameters. Table 10 showed that using the tuned parameters, the accuracy score is increased from 83.15% to 85.57%. The evaluation score in each class is presented in Table 9, and the confusion matrix showed in Figure 3.

Table 8. The comparison of evaluation score between DNN and DNN+Batch Normalization

Evaluation Score	DNN	DNN+BN
Accuracy	0.65368	0.83157
F1-Score	0.62568	0.82881
Precision	0.61076	0.82955
Recall	0.65368	0.83157
Kappa	0.47439	0.75629
Time (seconds)	115.9543	49.2367

Table 9. The comparison of evaluation score before tuning parameters and after tuning parameters

Evaluation Score	DNN+BN Before tuning parameters (N=50))	DNN+BN After tuning parameters (N=50)
Accuracy	0.83157	0.85578
F1-Score	0.82881	0.85348
Precision	0.82955	0.85298
Recall	0.83157	0.85578
Kappa	0.75629	0.79074
Time (seconds)	49.2367	250.255

Table 10. Each class evaluation score for the model

Class	precision	recall	f1-score	support
1	0.88	0.94	0.91	448
2	0.68	0.58	0.63	72
3	0.84	0.80	0.82	97
4	0.85	0.85	0.82	186
5	0.85	0.78	0.85	147
avg / total	0.85	0.86	0.85	950

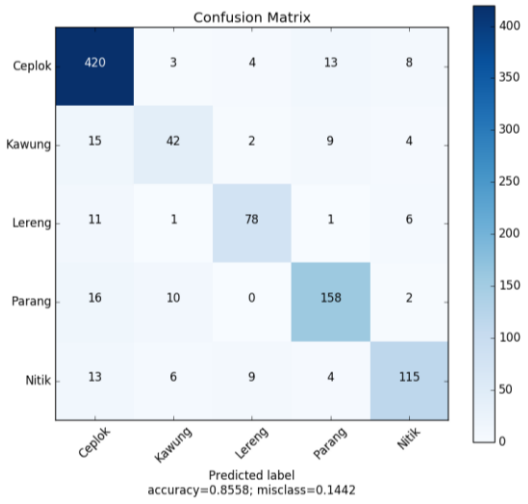

Figure 3. Confusion matrix

5. CONCLUSION

From the experiments in this study, the results showed that the feature extraction, selection, and reduction gave better accuracy than the raw dataset. The feature selection and reduction also reduce time complexity. The DNN+BN significantly improve the accuracy of the classification model from 65.36% to 83.15%. BN, as a regularisation, has successfully made the model more general, hence improve the accuracy of the model. The parameters are tuning improved accuracy from 83.15% to 85.57%. Parameters value that gave best accuracy are: batch_size = 20, epoch = 200, learning_rate = 0.001, optimizer = RMSprop, network weight initialization = Uniform, and neuron activation function = Softplus.

ACKNOWLEDGEMENTS

The research fund is supported by *Penelitian Terapan Unggulan Perguruan Tinggi* grant from Ministry of Research Technology and Higher Education, Republic Indonesia, 2019. We acknowledge the support and the encouragement for finalizing manuscript by Cultural and Education Attaché, the Embassy of the Republic Indonesia, London, United Kingdom, 2019

REFERENCES

[1] A. Wardhani, S. Sabana, and I. Adriati, "The Shift from Printed to Digital Magazine and Its Impact to Women Reader in Jakarta," *Procedia - Soc. Behav. Sci.*, vol. 184, pp. 310 – 314, 2015.
[2] H. Alwi, H. Lapoliwa, S. Darmowidjojo, and A. Moeliono, "Tata Bahasa Baku Bahasa Indonesia-Indonesian Standard Grammar," 3rd ed. Jakarta: Balai Pustaka, 2003.
[3] J. Helsing, "Batik the Traditional Fabric of Indonesia," Amazone Med, 2017.
[4] A. Wulandari, "Batik Nusantara (Makna Filosofi, Cara Pembuatan & Industri Batik)-Batik Nusantara (The Meaning of Philosophy, How to Manufacture & Manufacture Batik)," Yogyakarta: Andi Yogyakarta, 2011.
[5] H. R. Sanabila and M. Manurung, "Recognition of Batik Motifs using the Generalized Hough Transform," in *Proceedings of the International Conference on Advanced Computer Science and Information Systems, (ICACSIS)*, pp. 1–6, 2009.
[6] L. Rahadianti, R. Manurung, and A. Murni, "Clustering Batik Images based on Log-Gabor and Colour Histogram Features," in *Proceedings of the International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2009.
[7] I. Nurhaida, R. Manurung, and A. M. Arymurthy, "Performance Comparison Analysis Features Extraction Methods for Batik Recognition," in *2012 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 978–979, 2012.
[8] A. H. Rangkuti, R. B. Bahaweres, and A. Harjoko, "Batik Image Retrieval Based on Similarity of Shape and Texture Characteristics," in *International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 978–979, 2012.
[9] H. Rangkuti, A. Harjoko, and A. E. Putro, "Content Based Batik Image Retrieval," *J. Comput. Sci.*, vol. 10, no. 6, pp. 925–934, Jun. 2014.
[10] A. E. Minarno, Y. Munarko, F. Bimantoro, A. Kurniawardhani, and N. Suciati, "Batik image retrieval based on enhanced micro-structure descriptor," in *2014 Asia-Pacific Conference on Computer Aided System Engineering (APCASE), Bali,* 2014, vol. 1, pp. 65–70.
[11] A. E. Minarno, Y. Munarko, A. Kurniawardhani, F. Bimantoro, and N. Suciati, "Texture Feature Extraction Using Co-Occurrence Matrices of Sub-Band Image For Batik Image Classification," in *2nd International Conference on Information and Communication Technology (ICoICT),* 2014.
[12] I. Nurhaida, A. Noviyanto, M. Manurung, and A. M. Arymurthi, "Automatic Indonesian's Batik Pattern Recognition using SIFT Approach," *Procedia Comput. Sci.*, vol. 59, pp. 567–576, 2015.
[13] H. Fahmi, R. A. M. Zen, H. R. Sanabila, I. Nurhaida, and A. M. Arymurthy, "Feature Selection and Reduction for Batik Image Retrieval," in *Proceedings of the Fifth International Conference on Network, Communication and Computing (ICNCC)*, 2016.
[14] I. Nurhaida, H. Wei, R. A. M. Zen, R. Manurung, and A. M. Arymurthy, "Texture Fusion for Batik Motif Retrieval System," *Int. J. Electr. Comput. Eng.*, vol. 6, no. 6, pp. 3174–3187, 2016.
[15] Y. Gultom, A. M. Arymurthy, and R. J. Masikome, "Batik Classification using Deep Convolutional Network Transfer Learning," *J. Ilmu Komput. dan Inf.*, vol. 11, no. 2, p. 59, 2018.
[16] M. Najafabadi, F. Villanustre, T. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep Learning Applications and Challenges in Big Data Analytics," *J. Big Data*, vol. 2, no. 1, 2015.
[17] Y. Li, H. Zhang, X. Xue, Y. Jiang, and Q. Shen, "Deep Learning for Remote Sensing Image Classification: A Survey," *WIREs Data Min. Knowl. Discov.*, vol. 1264, no. 8, pp. 1–17, 2018.
[18] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the Importance of Initialization and Momentum in Deep Learning," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 1139–1147.
[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, 2012, pp. 1–9.

[20] J. Lamos-Sweeney, "Deep Learning Using Genetic Algorithms," Rochester Institute of Technology, Rochester, 2012.
[21] L. M. R. Rere, M. I. Fanany, and A. M. Arymurthy, "Simulated Annealing Algorithm for Deep Learning," *Procedia Comput. Sci.*, vol. 72, pp. 137–144, 2015.
[22] V. Ayumi, L. M. R. Rere, M. I. Fanany, and A. M. Arymurthy, "Optimization of Convolutional Neural Network using Microcanonical Annealing Algorithm," in *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2016, pp. 506–511.
[23] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Sci. J.*, vol. 313, no. 5786, pp. 504–507, 2006.
[24] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML)*, 2014, pp. 448–456.
[25] I. H. Ikasari, V. Ayumi, M. I. Fanany, and S. Mulyono, "Multiple regularizations deep learning for paddy growth stages classification from LANDSAT-8," in *International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2016, pp. 512–517.
[26] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, "Batch normalized recurrent neural networks," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 2657–2661.